

Re: Writing bulletproof code

Source: <http://coding.derkeiler.com/Archive/General/comp.programming/2004-01/0447.html>

From: Richard Heathfield (*dontmail_at_address.co.uk.invalid*)

Date: 01/05/04

Date: Mon, 5 Jan 2004 22:37:59 +0000 (UTC)

Malcolm wrote:

>
> *"Richard Heathfield" <dontmail@address.co.uk.invalid> wrote in*
>> *On the contrary, I consider correctness to be more important than*
>> *efficiency. If you think that I don't, then you have misunderstood me.*
>>
> *And you call yourself a games programmer?*

I'm not quite sure how to take that. :-)

> *Sometimes correctness is paramount.*

It's always paramount, insofar as a program that doesn't do what it's specified to do is broken.

> *Normally we make some compromises*
> *(such as storing real variables in 8 bytes) in the interests of*
> *efficiency. In games we accept almost any quick and dirty result in the*
> *interests of efficiency. After all, at the end of the day we're only*
> *setting pixels on the screen. Who cares if that Laura is a pixel out?*

If the spec cares, we care. If I can take an example from the world of games programming, then...

To calculate the distance between the points (x1, y1) and (x2, y2) we can do this:

```
double dist(int x1, int y1, int x2, int y2)
{
    double x = (double)x1 - x2; /* the cast is to avoid theoretical overflow
problems */
    y = (double)y1 - y2;
    return sqrt(x * x + y * y);
}
```

but we can, if we choose, do this instead:

comp.programming: Re: Writing bulletproof code

```
double dist(int x1, int y1, int x2, int y2)
{
    double x = (double)x1 - x2;
    double y = (double)y1 - y2;
    double m = x;
    if(x < 0)
    {
        x = -x;
    }
    if(y < 0)
    {
        y = -y;
    }
    if(y < x)
    {
        m = y;
    }
    return x + y - m / 2;
}
```

(In practice, we'd use ints internally, and a right shift instead of a division if we knew the compiler wouldn't spot it. I only used doubles to avoid nit-picking about overflow.)

The first version is neither completely accurate nor completely precise, but it's as good as we're going to get without going to extraordinary lengths.

The second version is rather less accurate and indeed less precise than the first version. Nevertheless, if it's accurate *enough* for our purposes, then it remains correct. After all, as you say, it doesn't matter whether we calculate the distance as 103 pixels or 103.0397 pixels, if we only have pixel resolution anyway.

In other words, if we make a deliberate decision to calculate a distance that is "good enough", then – as long as that result /is/ good enough – it's correct, even if it isn't 100% precise. But if 100% precision is required, then "good enough for rock n' roll" doesn't cut it any more.

--

Richard Heathfield : binary@eton.powernet.co.uk
"Usenet is a strange place." - Dennis M Ritchie, 29 July 1999.
C FAQ: <http://www.eskimo.com/~scs/C-faq/top.html>
K&R answers, C books, etc: <http://users.powernet.co.uk/eton>